



Agent-based Programming in Perl

Guinevere Nell
guinevere.nell@gmail.com

A short history of agents

Agent-based programming, or multi-agent programming, can be used to model anything from Conway's Game of Life to molecular or ecological representations, or even social interactions such as marketplaces, rumor mills and politics. Its great versatility does not mean that it's difficult or complicated to learn.

Agent-based modeling probably originated with the Von Neumann machine. This theoretical machine would follow detailed instructions to create a copy of itself; after the addition of Stanislaw Ulam's idea that it can be represented as a collection of cells on a grid, this machine became the first cellular automata. John Conway created the well-known Game of Life, which existed in a virtual world in the form of a 2-dimensional checkerboard.

The birth of agent-based model as a model for social systems began with computer scientist Craig Reynold. He attempted to model the reality of biological agents which became known as "artificial life", a term coined by Christopher Langton.

Agents in Perl

Perl is an excellent language for agent-based programming because it is so easy to churn out code; it has the right tools and the fluid syntax and the get-on-with-it attitude. So, if you are new to agent-based programming, there is no better way to get started than to use Perl.

At its most basic, an agent-based program includes separate units—objects, forked processes, threads—that each act independently with distinct features or actions over time. These agents can then interact with each other, update a common set of variables, or simply each run in unison or in sequence, each

doing their own thing. What really distinguishes agent-based programming from other kinds is that it always produces distinct agents for some purpose, but the most exciting applications are those that ultimately allow the agents to interact in some form and even learn and grow and change.

I could fork a process for each agent, but that would be CPU intensive. So, how can I create such a program without forking or threading? Simple: loop through and create objects. Can they interact if they are run sequentially? Sure! They can use a common pool of goods for exchange; global variables (or a database) for environmental factors; delayed messaging, and so on. So why do I want to use agents?

Economics

In economics, a recurring problem is that models can over-simplify the world and it is sometimes hard to know whether a model is too simple. Assumptions of a static state in economics can lead to models which miss major consequences of policy changes. Recently, agent-based modeling is gained ground in economics. Industrial organization, labor markets, cooperation and game theory, and many other concentrations have all been areas of focus for agent-based modeling.

Economics is only one field where agent-based models are used – biology (molecular, evolutionary and ecological), information theory, political science and many other fields are all being investigated with agent models. In each of these fields the agent model allows for emergence of phenomena from the lower (micro) level of the social system where individuals act to the higher level (macro), which is society.